An LTI control toolbox - simplifying optimal feedback controller design

Maarten Verbandt, Jan Swevers, Goele Pipeleers

***Abstract*— This paper presents a novel LTI control toolbox, aiming at transparent optimal feedback controller design. It provides a compact and intuitive syntax to model the control configuration and to formulate performance specifications based on an $\mathcal{H}_\infty$ criterion. Its core contains state-of-the-art solvers which are able to handle unstable weights. Moreover, improper weighting functions can be handled and state measurements are exploited to reduce the controller order. On account of these aspects, the synthesized controllers require no post-processing. A comparison with Matlab's robust control toolbox shows a clear improvement regarding the expression of controller specifications and the computation time.**

I. INTRODUCTION

Imagining today's industry without feedback controllers is almost impossible. Typical control structures such as PID controllers are applied, possibly with additional filters to counteract resonances or anti-resonanses. Although these building blocks have proven themselves useful, state-of-the-art feedback controller design strategies have the potential to outperform classical controllers as they provide more degrees of freedom. Moreover, these advanced methods gain industrial relevance as performance demands keep on increasing. Take for instance motion control of wafer steppers. The continuous demand for both higher throughput and higher accuracy results in a more and more challenging controller design. In order to meet the expectations, every new generation becomes lighter. This causes flexible modes to shift towards the region where tight control is necessary, hereby complicating the controller design.

The $\mathcal{H}_\infty$ design framework seems to be a promising alternative to classical open loop shaping. It allows translating performance specifications directly to the closed loop. These can be either objectives, such as maximize bandwidth, or constraints such as a bound on the sensitivity to sensor noise. Despite their potential, practical difficulties such as the complexity of controllers and the required expert knowledge, are still overruling the benefits and hereby sustain the dominance of standard PID control in industry.

Acknowledgement The authors would like to thank M. Dhadamus for his implementation of mixedHinfsynMIMO.

This work has been carried out within the framework of projects Flanders Make SBO ROCSIS: Robust and Optimal Control of Systems of Interacting Subsystems and KU Leuven-BOF PFV/10/002 Centre of Excellence: Optimization in Engineering (OPTEC). This work also benefits from the Belgian Programme on Interuniversity Attraction Poles, initiated by the Belgian Federal Science Policy Office (DYSCO) and FWO G.0915.14: project G.0915.14 of the Research Foundation - Flanders (FWO - Flanders)

The authors are with the Faculty of Mechanical Engineering, Division PMA, KU Leuven, BE-3001 Leuven, Belgium maarten.verbandt@kuleuven.be

Traditionally, the difficulties regarding $\mathcal{H}_\infty$ design occur at two levels. First there is a lack of software support. Although Matlab's robust control toolbox contains a variety of functions, they are all limited to either unconstrained optimization or feasibility problems and require a high level of expertise from the user. Second, post-processing is required before the controller can be deployed, typically removing near pole-zero cancellations and dynamics located outside the control region. To overcome these difficulties, we have developed a Matlab LTI Control Toolbox that provides a clear interface to propose a control configuration and a set of requirements. The implementation of state-of-the-art controller synthesis tools allows the user to design various controllers such as controllers with integrators and reduced order controllers. In what follows, we synthesize reduced order $\mathcal{H}_\infty$ controllers by means of the LTI control toolbox which do not require post-processing. These are obtained by choosing unstable and improper weighting functions combined with a suitable solver. By detecting direct state measurements, the order of the controller can be reduced without losing performance.

This paper is organized as follows. Section II starts by introducing recent developments in $\mathcal{H}_\infty$ controller design which form the basis of the controller synthesis. The following sections discuss the new LTI control toolbox and provide a comparison with Matlab's robust control toolbox. This comparison is based on a mechatronic case study for which we synthesize both a classic single input controller and a reduced order multiple input controller. Section V concludes the paper.

II. $\mathcal{H}_\infty$ CLOSED LOOP SHAPING

Consider a controller design for which the closed loop has to meet the following requirements:

$$|\mathcal{T}_i(\omega)| \leqslant \left|\frac{1}{\mathcal{W}_i(\omega)}\right|, \quad \forall \omega \in \mathbb{R},\, i \in \{1, ..., N\} \qquad (1)$$

where $\mathcal{T}_i(\omega)$ represents the closed-loop transfer function on which the bound $\mathcal{W}_i^{-1}(\omega)$ is requested. The $\mathcal{H}_\infty$ framework provides a way to directly synthesize a controller that meets these constraints by means of an optimization problem. Traditional $\mathcal{H}_\infty$ methods only allow stacked objectives and the minimization of one $\mathcal{H}_\infty$ norm [1], for example Matlab's hinfsyn. Although it provides a way to obtain a stabilizing controller that more or less satisfies the constraints, this way of formulating the problem is far from transparent for a non-expert.

Recent advances in the field do not require the weighted transfer functions to be stacked [2]. Therefore objectives

and constraints can be rigorously separated, resulting in an optimization problem of the form:

$$\begin{aligned} \underset{K}{\text{minimize}} \quad & \sum_i \alpha_i \left\| \mathcal{W}_i \mathcal{T}_i \right\|_\infty \\ \text{subject to} \quad & \left\| \mathcal{W}_j \mathcal{T}_j \right\|_\infty \leqslant 1 \end{aligned} \tag{2}$$

The weights $\mathcal{W}_j$ implicitly limit the magnitude of the closed-loop transfer functions $\mathcal{T}_j$. This provides a way to restrict the peak sensitivity to improve damping of the closed-loop poles or to put a bound on the complementary sensitivity to ensure robustness. Since the design is formulated as an optimization problem, the remaining freedom in controller parameters can be exploited to minimize some function. This is in general expressed by a weighted sum over the several objectives, $\left\| \mathcal{W}_i \mathcal{T}_i \right\|_\infty$, by means of α_i. This is used to for instance minimize the actuator effort or maximize the bandwidth.

Although appealing, these $\mathcal{H}_\infty$ methods suffer from impracticalities due to mathematical reasons, as described in the following paragraphs.

A. Unstable weights: enforcing integrators in the controller

Typically only stable weights, $\mathcal{W}$, can be applied in (1) and (2) since the generalized plant has to remain stabilizable. This prohibits including pure integrators in the weights in order to enforce integrators in the controller. The usual workaround consists of transforming the desired weight into a stable alternative with low frequency poles. This results in a controller that displays integrator-like dynamics, which are converted to a real integrator in a post-processing step. Recently, [3] proposed a way to deal with unstable weights by enforcing a pole-zero cancellation using the controller's dynamics. This allows the user to directly create an integrator in the controller avoiding post-processing.

B. Improper weights: enforcing roll-off in the closed-loop

Traditional methods rely on the state-space description of the generalized plant. Since the state-space description only exists for proper plants, weights are typically chosen to be proper. Improper weights can however be desirable as they enforce roll-off on the weighted channel. In order to mimic their result, the weight is augmented with high frequency poles, rendering the weight proper. However this results in an needless increase of the controller order and additional higher order dynamics that should be removed in an additional post-processing step.

If the channel, $\mathcal{T}$, has a relative degree, n, an improper weight $\mathcal{W}$ up to relative degree $-n$ will not result in an improper weighted channel, $\mathcal{WT}$. In this case, traditional methods offer a solution.

However, if the weighted channel becomes improper, one has to resort to the descriptor form:

$$\begin{aligned} E\dot{x} &= Ax + Bu \\ z &= Cx + Du \end{aligned} \tag{3}$$

In this case E is rank deficient. A promising approach to handle a rank deficient E is proposed in [4] where a pole-zero cancellation at infinity is enforced by inserting the required dynamics in the controller. This results in an impulsive-free descriptor system for which $\mathcal{H}_\infty$ design methods exist [5].

C. Order reduction

Classic $\mathcal{H}_\infty$ design methods synthesize a controller of the same order as the generalized plant. This assumption allows the controller synthesis problem to be reformulated as a convex optimization problem [6]. However, since a $\mathcal{H}_\infty$ controller can be seen as the combination of a state estimator and state feedback [7], direct state measurements can be exploited to reduce the order of the controller. Such technique is explained in [8].

III. LTI CONTROL TOOLBOX

$\mathcal{H}_\infty$ controller synthesis typically follows a fixed procedure. First the plant and the control configuration are modeled. Second the specifications are formulated as an optimization problem, e.g. as suggested by Eq. (2). The combination of systems and weights yields the generalized plant which is then used by a suitable solver to synthesize a controller.

The developed LTI control toolbox[1] supports the user in these modeling steps by providing a clear syntax to formulate both the control configuration and requirements in an intuitive way. Behind the scenes, the corresponding generalized plant is constructed and the design problem is transferred to an appropriate solver.

The following subsections show the structure of the toolbox and the syntax to define the control configuration and the control requirements formulation.

A. Control configuration

The control configuration consists of all dynamic systems and the connections between their inputs and outputs. In order to easily connect all systems, a signal-based Matlab syntax has been developed. As an example, the plant in Fig. 1 is described by Code example 1.

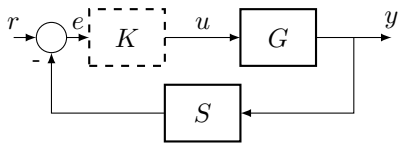

Fig. 1: Schematic representation of the example control configuration. G and S are dynamic systems which together form the plant. K indicates the controller which eventually closes the loop.

```
1  G = LTIsys(G); %make G.in and G.out available
2  S = LTIsys(S);
3
4  lti_begin()
5      signal r
6
7      u = G.in;            %define u as G's input
```

[1]The developed LTI control toolbox is freely available on github: https://github.com/maartenverbandt/lti_toolbox

```
8       y = G.out;          %define y as G's output
9       e = r - S.out;      %define the error
10      S.in == y;          %connect S to G
11
12      K.in = e;           %set controller input
13      K.out = u;          %set controller output
14  lti_end
```

Code example 1: LTI control toolbox code to describe the control configuration depicted in Fig. 1.

In order to make G and S available to the toolbox, they need input and output signals assigned to them, which is done in lines 1 to 2. In between the *lti_begin* and *lti_end*, the configuration is defined. Since r is an external reference, it is declared as a new signal. Next u and y are declared as convenience variables for the input and output of $G.e$, the error, is declared as r minus the output of S. Finally, G and S are connected by making $S.in$ and y equal. Lines 12 and 13 specify the inputs and outputs of the eventual controller.

Although this syntax is provided as a tool to ease the declaration of the control configuration, it can also be used to interconnect a series of systems in an arbitrary manner to obtain a dynamic description of the whole.

B. Control requirements

The control requirements reflect the wishes of the designer regarding the controller's performance. In an $\mathcal{H}_\infty$ framework, this is done by specifying bounds, $\mathcal{W}_i$, on the closed-loop transfer functions, $\mathcal{T}_i$, which may also be minimized. Since Eq. (1) can be rewritten as:

$$\|\mathcal{W}_i\mathcal{T}_i\|_\infty \leqslant 1 \tag{4}$$

$\mathcal{W}_i$ can be regarded as a weight for the closed-loop transfer function. The LTI control toolbox provides a natural way of stating these specifications in terms of weighted transfer functions. Consider as a design problem the case where the actuator effort needs to be minimized given a minimal bandwidth w.r.t. tracking error rejection. These specifications can be reformulated in terms of two weights, $\mathcal{W}_U$ and $\mathcal{W}_S$, and the following optimization problem:

$$\begin{aligned} \underset{K}{\text{minimize}} \quad & \left\| \mathcal{W}_U \frac{u}{r} \right\|_\infty \\ \text{subject to} \quad & \left\| \mathcal{W}_S \frac{e}{r} \right\|_\infty \leqslant 1 \end{aligned} \tag{5}$$

Code example 2 shows the equivalent declaration using the LTI control toolbox. The design starts by declaring $\mathcal{W}_U$ and $\mathcal{W}_S$. The class *Weight* has been implemented to provide standard weights such as low and high frequency roll-off weights or a DC weight. The actual control problem specification is done in the *ctrl_begin() .. ctrl_end()* environment. It accepts *minimize* and *subject to* to specify the objective and constraints respectively. The syntax allows an almost exact copy of the previously stated optimization problem (Eq. 5).

```
1  % unity weight
2  WU = LTIsys(Weight.DC(0));
3  % order 1 unstable weight with w0 Hz BW
4  WS = LTIsys(Weight.LF(w0,1));
5
6  lti_begin()
7      %% ... %% plant declaration
8
9      ctrl_begin('my_controller')
10         minimize(WU*(u/r))
11         subject to
12             WS*(e/r) <= 1
13     ctrl_end
14 lti_end
```

Code example 2: LTI control toolbox code to design the controller as described by problem (5).

In order to compare multiple controller designs at once, the toolbox allows multiple *ctrl_begin() .. ctrl_end()* statements. This results in a set of controllers for the same plant whose performance is readily compared.

The toolbox contains multiple LMI-based solvers all handling different convex problems. mixedHinfsynMIMO is our own implementation of state-of-the-art functionality as touched upon in section II. It can handle unstable weights and synthesizes reduced order controllers by means of additional state feedback.

C. Structure

Fig. 2 displays the structure of the LTI control toolbox. The structure rougly represents the aforementioned design steps: the configuration, i.e. the modeling of the plant, and the requirement specification, i.e. formulating an optimization problem using weighted transfer functions. The configuration can be regarded as a plant building problem. Several dynamic systems and adders are being connected through a set of signals. These are all stored in one class which can be translated to a model of the entire plant. The requirements class contains a series of weighted transfer function norms which reflect the design specifications of the user. Combined with the configuration object, the generalized plant can be constructed. Each control specification object also contains a solver object that processes the objectives and constraints and calls a suitable solver.

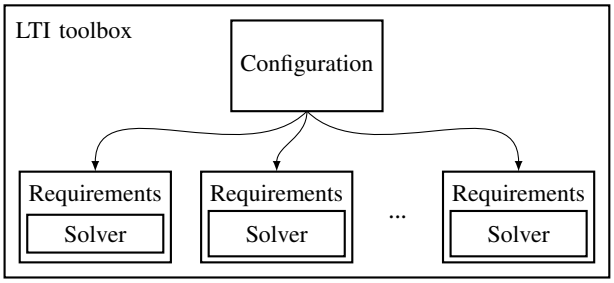

Fig. 2: Schematic representation of the LTI control toolbox's internal structure.

IV. COMPARISON WITH MATLAB ROBUST CONTROL

This section provides a comparison of the newly developed LTI control toolbox with the standard robust control toolbox in Matlab. This comparison is based on a mechatronic case study[2] which is outlined in a first subsection. The second

[2]This case study is distributed along with the toolbox and can be found as part of the standalone examples under ECC2016.

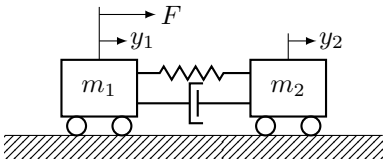

Fig. 3: Schematic representation of the mechatronic system to be controlled. In between the two masses sits a spring-damper system. Both positions are being measured while only the first mass is actively driven by the input F.

subsection demonstrates the simplicity which the control configuration and the optimization problem are defined with. The third subsection compares the performance of the available solvers.

A. Mechatronic case study

The considered mechatronic system is depicted in Fig. 3. A spring-damper system connects two masses of which the first is driven by an input force, F. The positions of both masses are being measured, y_1 and y_2. This results in the 2-by-1 system G, whose transfer function is depicted in Fig. 4. Furthermore the multiplicative uncertainty bound on the model is known and characterized by $\mathcal{W}_T$.

The control configuration is shown in Fig. 5. First we consider a single input control design based on e (collocated control configuration), indicated by the solid lines. Second a 3-input controller is synthesized using y_1 and y_2 additionally (dashed lines).

In what follows, the transfer function $r \mapsto e$ will be refered to as the sensitivity, S, and $r \mapsto y_1$ as the complementary sensitivity, T. The purpose of the design is to achieve optimal reference tracking of the first mass without a steady-state positioning error, disregarding the response of the second mass. This requirement dictates an integrator in the controller. Since the first output of G already contains two integrators, a weight on the sensitivity, $\mathcal{W}_S$, with triple integral action is required to enforce an extra integrator in the open loop. In order to prevent a large overshoot on the step response the sensitivity should stay below $8dB$. This puts a constraint on the peak of the sensitivity function, $\mathcal{M}_S$. Robustness is taken into account via the multiplicative uncertainty w.r.t. the first output. This is done by demanding the complementary sensitivity to stay below the uncertainty bound $\mathcal{W}_T$. Fig. 6 depicts the selected weights. Dashed lines indicate the classical weights which are proper and stable. The solid lines represent their desired counterparts which are traditionally avoided since an unstabilizable or improper plant cannot be dealt with by the robust control toolbox.

B. Controller design

Reformulating the previously stated requirements as an $\mathcal{H}_\infty$ problem results in the following:

$$\begin{aligned} \underset{K}{\text{minimize}} \quad & \left\|\mathcal{W}_S S\right\|_\infty \\ \text{subject to} \quad & \left\|\mathcal{M}_S S\right\|_\infty \leqslant 1 \\ & \left\|\mathcal{W}_T S\right\|_\infty \leqslant 1 \end{aligned} \tag{6}$$

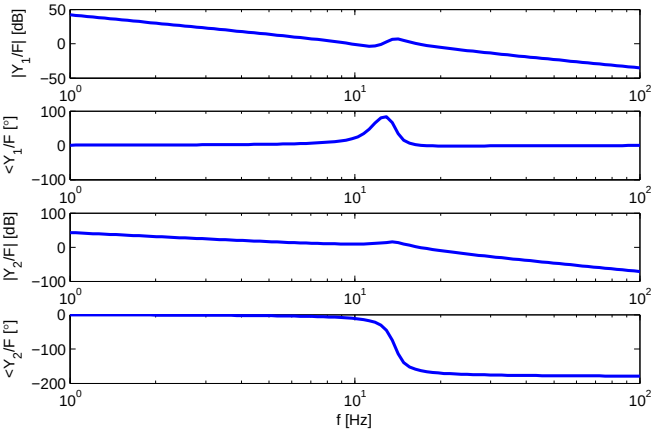

Fig. 4: Bode diagram of the plant, G.

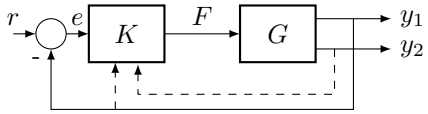

Fig. 5: Schematic of the control loop for both the single and multi input configuration. The dashed lines indicate the additional measurements being fed back to the controller.

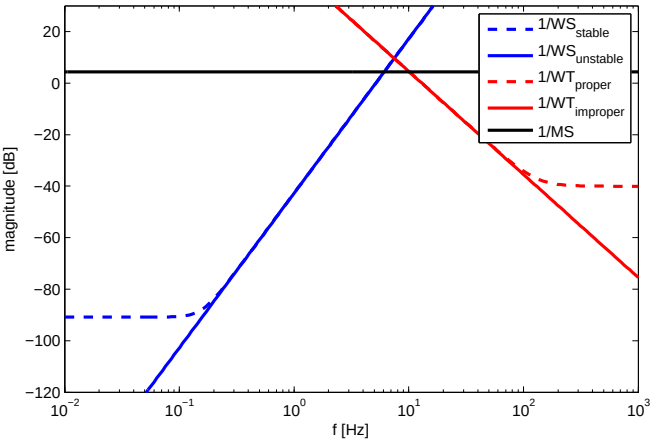

Fig. 6: Selected weights for the $\mathcal{H}_\infty$ controller design. Dashed lines indicate classic weighting functions whereas the solid lines mark their unstable and improper counterparts. The maximum sensitivity weight remains unchanged.

Both the sensitivity and complementary sensitivity function are bounded by respectively $\mathcal{M}_S$ and $\mathcal{W}_T$. The bandwidth is maximized by minimizing the infinity-norm of the weighted sensitivity function $\mathcal{W}_S S$.

Using the LTI control toolbox, the required code looks something like:

```
1  G = LTIsys(plant);
2  WS = Weight.LF(5,3);
3  MS = Weight.DC(8);
4  WT = Weight.HF(13,2);
5
6  lti_begin()
7      signal r
8
9      F = G.in;
```

```
10      y = G.out;     %2x1 signal
11      e = r - y(1); %tracking error of y1
12
13      K.in = e; %[e;y] for additional state fb
14      K.out = F;
15
16      ctrl_begin('controller',options)
17      minimize(WS*(e/r))
18      subject to
19          MS*(e/r) <= 1
20          WT*(y(1)/r) <= 1
21  ctrl_end
22  lti_end
```

Code example 3: LTI control toolbox code to synthesize the controller as stated in subsection IV-A. The first part describes the control configuration whereas the second part specifies the design requirements.

This design can be redone using Matlab's robust control toolbox. This results in Code example 4.

```
1   Ny = 1;
2   Nx = 9; %Nx is 5 for state fb
3   Nu = 1; %Nu is 3 for state fb
4   K = ltiblock.ss('K',Nx,Ny,Nu);
5   K.u = 'e'; %{e;y;ya} for additional state fb
6   K.y = 'u';
7
8   WS.u = 'e';  WS.y = 'z1';
9   MS.u = 'e';  MS.y = 'z2';
10  WT.u = 'y'; WT.y = 'z3';
11  G.u = 'u'; G.y = {'y';'ya'};
12
13  sum_e = sumblk('e=r-y');
14
15  P = connect(G,K,WS,MS,WT,sum_e,...
16      {'r'},{'z1','z2','z3'});
17
18  ReqWS = TuningGoal.Gain('r','z1',1);
19  ReqMS = TuningGoal.Gain('r','z2',1);
20  ReqWT = TuningGoal.Gain('r','z3',1);
21
22  systuneOptions('RandomStart',5);
23  [CL] = systune(P,[ReqWS],[ReqMS;ReqWT]);
```

Code example 4: Robust control toolbox code to synthesize the controller as stated in subsection IV-A. There is no clear separation between the control configuration and the requirements.

Two main differences catch the eye when comparing Code example 3 to 4. Whereas the robust control toolbox connects systems via the systems' input and output names, the novel LTI control toolbox uses a list of input and output signals. The latter has the advantage of providing a shorter way of notation. Also the user is not obliged to specify signal names in order to interconnect systems. Systems can be interconnected directly via the systems' signals e.g. G.in and G.out.

The LTI control toolbox also allows a natural description of the control requirements, lowering the necessary level of expertise of the user. Therefore this toolbox clearly simplifies the design procedure compared to the robust control toolbox.

C. Evaluation of the available solvers

Matlab's robust control toolbox contains various solvers aiming at $\mathcal{H}_\infty$ controller design. Without a doubt hinfsyn is the most famous. More advanced methods such as hinfstruct and systune allow the user to design structured fixed order controllers. Since they rely on a non-convex formulation of the $\mathcal{H}_\infty$ design problem [9], they risk getting stuck in a local optimum.

As a first example, the closed-loop performance obtained with various solvers for the classical full order design is shown in Fig. 7. In this case, hinfstruct achieves a performance comparable to mixedHinfsynMIMO. Due to its non-convexity, systune gets stuck in a local optimum, resulting in a lower bandwidth and poor compensation of the anti-resonance-resonance pair.

Since mixedHinfsynMIMO can handle the unstable sensitivity weight, it is possible to directly obtain the desired integrator in the controller (Fig. 8). This is not the case when using the built-in Matlab tools, resulting in the differentiator-like characteristics at low frequencies.

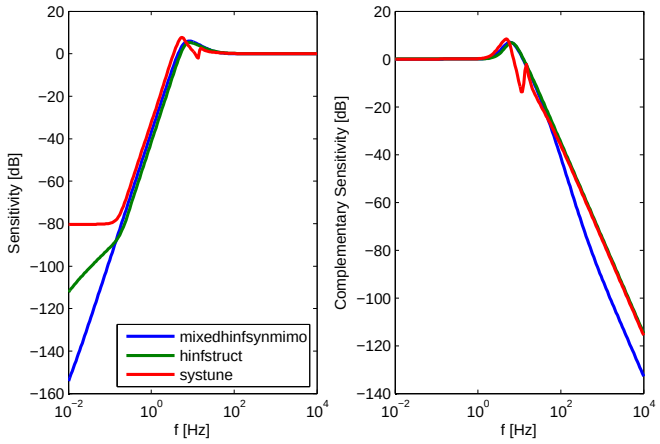

Fig. 7: Comparison of the closed loop performance when using mixedHinfsynMIMO, hinfstruct and systune for the single input controller design.

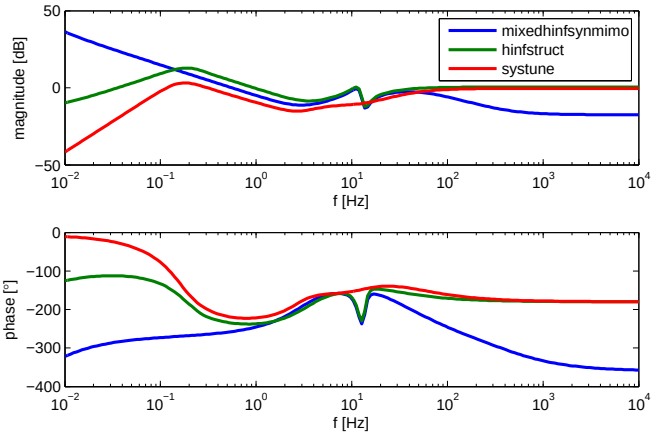

Fig. 8: Comparison of the controller when using mixedHinfsynMIMO, hinfstruct and systune for the single input controller design.

Important to note is that contrary to mixedHinfsynMIMO and systune, hinfstruct minimizes the stacked norm of all

exogenous outputs. This means that obtaining optimal performance given a set of hard constraints is not possible in a direct way. A workaround consists of altering the weights based on mixedHinfsyn's solution. This results in a controller satisfying both the constraints and optimality conditions, such that it is comparable to the other controllers.

Second, we design a controller for the mutli input configuration. Based on Fig. 9, all controllers show similar performance. Since Matlab's robust control toolbox does not allow unstable weights, a pure integrator is not realizable resulting in a non-zero sensitivity at $0Hz$. With the single input controller design, it was still easy to move the controller's low frequency poles to $0Hz$ in order to obtain the integrator. This however is no longer the case for the multiple input controller, emphasizing the relevance of these unstable weights. Furthermore, the explicit partial state feedback allows the synthesis of a reduced order controller, going from a 7th to 5th order controller without loss of performance.

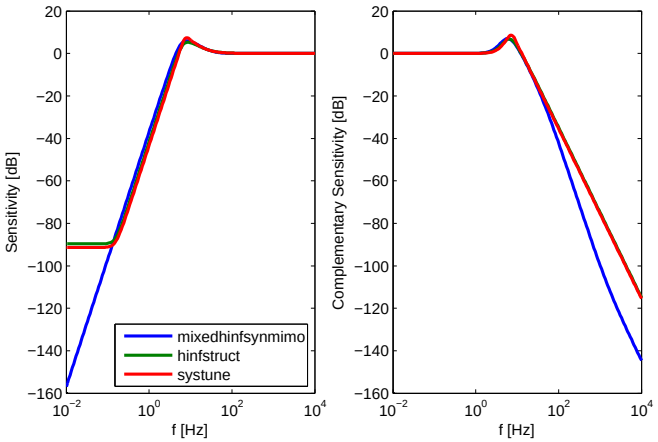

Fig. 9: Comparison of the closed loop performance when using mixedHinfsynMIMO, hinfstruct and systune for the multi input controller design.

Table I lists the controller synthesis time and the total time which is required to compute the controller and the closed-loop transfer function. Thanks to the convex reformulation, mixedHinfsynMIMO obtains a solution in a considerably shorter time than the built-in Matlab tools. It is also possible that the LMI solver benefits from the fact that the order of the system is rather low, so one should be careful when interpreting these results.

What also catches the eye is that although hinfstruct and systune both rely on the same non-smooth optimization methods, solving the constrained optimization problem with systune is without a doubt faster than solving the stacked optimization problem with hinfstruct.

V. CONCLUSION

Several issues prevent $\mathcal{H}_\infty$ controller design to be applied in industry. Obvious reasons are the lack of software support and the need for post-processing. The developed LTI control toolbox simplifies the control problem formulation and synthesis considerably. A comparison of the LTI control toolbox with Matlab's robust control toolbox shows that the former is more intuitive and that its routines handle more complex problems such as constrained optimization problems with unstable and improper weights. Moreover, the considered case study shows the LTI control toolbox's potential to exceed the robust control toolbox in speed. By means of the LTI control toolbox, it is possible to synthesize a reduced order controller that is directly suited for operation.

single input	mHinfMIMO	hinfstruct	systune
synthesis time [s]	1.06	12.42	6.31
total time [s]	1.49	12.51	6.33
multi input	mHinfMIMO	hinfstruct	systune
synthesis time [s]	0.24	13.34	6.82
total time [s]	0.66	13.38	6.86

TABLE I: Computation times for the single and multi input controller design for the tested solvers.

REFERENCES

[1] T. Iwasaki and R. E. Skelton, "All controllers for the general $\mathcal{H}_\infty$ control problem: LMI existence conditions and state space formulas," *Automatica, Vol 30, No 8*, 1994.

[2] C. Scherer, P. Gahinet, and M. Chilali, "Multiobjective output-feedback control via LMI optimization," *Automatic Control, IEEE Transactions on*, vol. 42, no. 7, pp. 896–911, 1997.

[3] H. Köroğlu and C. W. Scherer, "Generalized asymptotic regulation with guaranteed $\mathcal{H}_\infty$ performance: An LMI solution," *Automatica*, vol. 45, no. 3, pp. 823–829, 2009.

[4] Y. Feng and Z. Su, "$\mathcal{H}_\infty$ control with output weights for descriptor systems," *Proceedings of the 33rd Chinese Control Conference*, 2014.

[5] I. Masubuchi, "Output feedback controller synthesis for descriptor systems sastifying closed-loop dissiptivity," *Automatica,Vol 43*, 2007.

[6] C. M. Agulhari, R. C. L. F. Oliveira, and P. L. D. Peres, "LMI relaxations for reduced-order robust $\mathcal{H}_\infty$ control of continuous-time uncertain linear systems," *IEEE Transactions on Automatic Control*, vol. 57, no. 6, pp. 1532–1537, 2012.

[7] S. Skogestad and I. Postlethwaite, *Multivariable feedback control: analysis and design*, vol. 2. Wiley New York, 2007.

[8] T. Asai and S. Hara, "Convex parametrization of reduced order controllers for a class of problems under partial state measurements," *Proceedings of the 36th IEEE Conference onf Decision and Control*, 1997.

[9] P. Apkarian and D. Noll, "Nonsmooth $\mathcal{H}_\infty$ synthesis," *Automatic Control, IEEE Transactions on*, vol. 51, no. 1, pp. 71–86, 2006.